Rapport de RODD

Première Partie - Protection de la biodiversité

Adrien BLASSIAU
Corentin JUVIGNY

1er février 2020

Table des matières

Introduction

Ce rendu regroupe l'ensemble des rapports associés aux 4 projets réalisés dans le cadre de la première partie de l'UE **Recherche Opérationnelle et Développement Durable** (RODD) encadrée par **Amélie Lambert** :

— un projet sur la sélection de réserves naturelles.

— un projet sur la maîtrise des effets néfastes engendrés par la fragmentation du paysage.

— un projet sur la protection de la diversité génétique.

— un projet sur l'exploitation durable de la forêt.

Nous présenterons les **modèles** ainsi que l'**analyse des résultats obtenus** sur les intances fournies et des instances générées aléatoirement.

1 Projet 1 - Sélection de réserves naturelles

1.1 Présentation du problème

On souhaite assurer à chaque espèce ou site menacés un espace ou son avenir est garanti.
Pour cela, on s'intéresse à un **ensemble d'espèces à protéger** $\mathrm{E} = \{e_1, e_2, \ldots, e_p\}$ vivant sur un **ensemble de parcelles** $\mathrm{S} = \{s_1, s_2, \ldots, s_n\}$ répartis sur un territoire.

L'**objectif** est de déterminer un sous-ensemble de parcelles de coût minimal et tel que, pour chaque espèce e_k, la probabilité de présence dans la réserve, c'est-à-dire dans ce sous-ensemble de parcelles, soit supérieure ou égale à une valeur donnée α_k.

1.2 Modélisation du problème

On obtient le programme non linéaire suivant ;

$$\min \quad z = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^{p} c_{ij}$$

$$\begin{aligned}
\text{s.c.} \quad & \sum_{i=1}^{m} x_{i1}^{c} = 0 && && \textbf{(C1)} \\
& \sum_{i=1}^{m} x_{in}^{c} = 0 && && \textbf{(C2)} \\
& \sum_{j=1}^{n} x_{1j}^{c} = 0 && && \textbf{(C3)} \\
& \sum_{j=1}^{n} x_{mj}^{c} = 0 && && \textbf{(C4)} \\
& 9x_{ij}^{c} \leq \sum_{a=i-1}^{i+1} \sum_{b=i-1}^{i+1} x_{ab}^{p} && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!] && \textbf{(C5)} \\
& 1 - \prod_{i=1}^{m} \prod_{j=1}^{n} 1 - p_{kij} x_{ij}^{c} \geq \alpha_k && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![1;p]\!] && \textbf{(C6)} \\
& 1 - \prod_{i=1}^{m} \prod_{j=1}^{n} 1 - p_{kij} x_{ij}^{p} \geq \alpha_k && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![p+1;p+q]\!] && \textbf{(C7)} \\
& x_{ij}^{c} \in [\![0;1]\!] && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!] && \\
& x_{ij}^{p} \in [\![0;1]\!] && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!] && \\
& \alpha_k \in [0,1] && \forall k \in [\![1;p+q]\!] && \\
& p_{ijk} \in [0,1] && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![1;p+q]\!] &&
\end{aligned}$$

On obtient le programme linéaire suivant, après linéarisation par le logarithme du programme précédent :

$$\min \quad z = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}^{p} c_{ij}$$

$$
\begin{aligned}
\text{s.c.} \quad & \sum_{i=1}^{m} x_{i1}^{c} = 0 && && \textbf{(C1)}\\
& \sum_{i=1}^{m} x_{in}^{c} = 0 && && \textbf{(C2)}\\
& \sum_{j=1}^{n} x_{1j}^{c} = 0 && && \textbf{(C3)}\\
& \sum_{j=1}^{n} x_{mj}^{c} = 0 && && \textbf{(C4)}\\
& 9x_{ij}^{c} \leq \sum_{a=i-1}^{i+1}\sum_{b=i-1}^{i+1} x_{ab}^{p} && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!] && \textbf{(C5)}\\
& \sum_{i=1}^{m}\sum_{j=1}^{n} \log(1-p_{kij})x_{ij}^{c} \leq \log(1-\alpha_k) && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![1;p]\!] && \textbf{(C6)}\\
& \sum_{i=1}^{m}\sum_{j=1}^{n} \log(1-p_{kij})x_{ij}^{p} \leq \log(1-\alpha_k) && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![p+1;p+q]\!] && \textbf{(C7)}\\
& x_{ij}^{c} \in [\![0;1]\!] && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!] &&\\
& x_{ij}^{p} \in [\![0;1]\!] && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!] &&\\
& \alpha_k \in [0,1] && \forall k \in [\![1;p+q]\!] &&\\
& p_{ijk} \in [0,1] && \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!], \quad \forall k \in [\![1;p+q]\!] &&
\end{aligned}
$$

Ce **programme linéaire en variable 0-1** modélise le problème de la détermination d'une réserve optimale avec zone centrale et zone tampon. On utilise **deux variables de décision** :

— x_{ij}^{p} prend 1 si la case de coordonnées (i,j) est **protégée**, 0 sinon.

— x_{ij}^{c} prend 1 si la case de coordonnées (i,j) est **centrale**, 0 sinon.

Les **contraintes** ont les significations suivantes :

— Les contraintes **C1** à **C4** formalise le fait que les cases bordant la zone d'étude **ne peuvent pas être centrales**.

— La contrainte **C5** formalise le fait que pour être **centrale**, une case **doit être entourée par 8 cases protégées**.

— La contrainte **C6** formalise le fait que la probabilité de présence d'une **espèce en danger** doit être **supérieure à un seuil fixé**.

— La contrainte **C7** formalise le fait que la probabilité de présence d'une **espèce commune** doit être **supérieure à un seuil fixé**.

1.3 Résolution du problème

On résout le problème en considérant 3 espèces rares numérotées de 1 à 3 et trois espèces communes numérotées de 4 à 6. On utilise la matrice des coûts du Tableau 1 et on étudie les 4 cas suivants :

— Cas n°1 : $\alpha_k = 0.5$ $(k = 1, \ldots, 6)$

— Cas n°2 : $\alpha_k = 0.9$ $(k = 1, \ldots, 3)$ et $\alpha_k = 0.5$ $(k = 4, \ldots, 6)$

— Cas n°3 : $\alpha_k = 0.5$ $(k = 1, \ldots, 3)$ et $\alpha_k = 0.9$ $(k = 4, \ldots, 6)$

— Cas n°4 : $\alpha_k = 0.8$ $(k = 1, \ldots, 3)$ et $\alpha_k = 0.6$ $(k = 4, \ldots, 6)$

6	6	6	4	4	4	4	8	8	8
6	6	6	4	4	4	4	8	8	8
6	6	6	4	4	4	4	8	8	8
5	5	5	3	3	3	3	7	7	7
5	5	5	3	3	3	3	7	7	7
5	5	5	3	3	3	3	7	7	7
5	5	5	3	3	3	3	7	7	7
4	4	4	6	6	6	6	5	5	5
4	4	4	6	6	6	6	5	5	5
4	4	4	6	6	6	6	5	5	5

TABLE 1 – Matrice des coûts utilisées

On obtient les résultats visibles sur le Tableau 2, le Tableau 3, le Tableau 4 et le Tableau 5 qui **correspondent aux résultats attendus**. Un récapitulatif des données voulues est donné dans le Tableau 6.

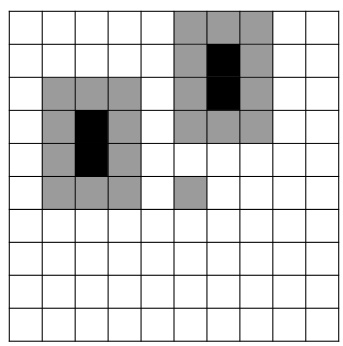

TABLE 2 – Résultats obtenus pour le **Cas n°1** : **coût = 119** et **probabilités de survies : (0.58, 0.52, 0.64, 0.86176, 0.52, 0.755)**.

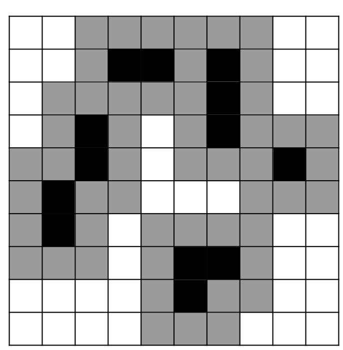

TABLE 3 – Résultats obtenus pour le **Cas n°2** : **coût = 327** et **probabilités de survies : (0.915328, 0.90784, 0.91936, 0.980491571, 0.892, 0.9814785)**.

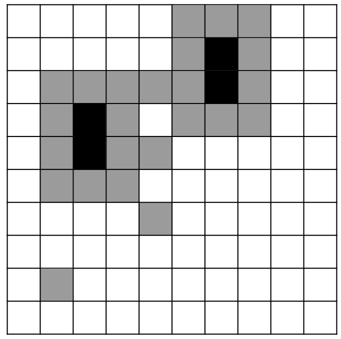

TABLE 4 – Résultats obtenus pour le **Cas n°3** : **coût = 130** et **probabilités de survies : (0.58, 0.52, 0.64, 0.9336448, 0.91, 0.9265)**.

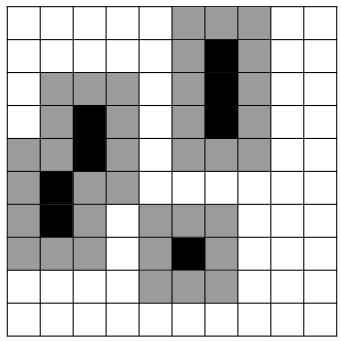

TABLE 5 – Résultats obtenus pour le **Cas n°4** : **coût = 211** et **probabilités de survies : (0.8236, 0.808, 0.82, 0.972130816, 0.784, 0.8775)**.

	temps (s)	nombre de noeuds	coût	probabilité de survie
Cas n°1	0,09	0	119	(0.58, 0.52, 0.64, 0.86176, 0.52, 0.755)
Cas n°2	0,01	0	327	(0.915328, 0.90784,0.91936, 0.980491571, 0.892, 0.9814785)
Cas n°3	0,12	0	130	(0.58, 0.52, 0.64,0.9336448, 0.91, 0.9265)
Cas n°4	0,06	0	211	(0.8236, 0.808, 0.82, 0.972130816, 0.784, 0.8775)

TABLE 6 – Résultats obtenus pour les différents cas demandés.

1.4 Étude de l'impact de la taille d'une instance sur le temps moyen de calcul

On étude l'**impact de la taille d'une instance sur le temps moyen de calcul**. Pour cela on crée des instances de manière aléatoire avec le générateur d'instance `generate.py` en suivant les règles suivantes :

— Les **coûts** sont choisis **entre 0 et 10 compris**.

— Les **probabilités** sont choisies **uniformément entre 0 et 1**.

— Les α sont choisies **uniformément entre 0 et 1**.

— On choisit **3** espèces **communes** et **3** espèces en **danger**.

— On fait varier les tailles de 10 en 10 en gardant $m = n$ en s'arrêtant à 100.

Notre machine de tests dispose de **8 GO de ram** et d'un **processeur i5**. Enfin, pour chaque taille d'instance, on effectue **10 expériences**. On s'intéresse donc au temps **moyen** (Figure 1) ainsi qu'à la répartition des valeurs obtenues (Figure 3) en fonction de la taille de l'instance.

On remarque que **plus la taille des instances est importante, plus le temps de calcul en moyenne est élevé et plus les écarts de temps de calcul sont importants**. L'évolution semble **exponentielle**. En passant au log (Figure 2), on observe que l'évolution est presque linéaire. Il aurait fallut faire plus d'expériences pour avoir un résultat plus précis. On peut tout de même imaginer que le problème est **difficile**.

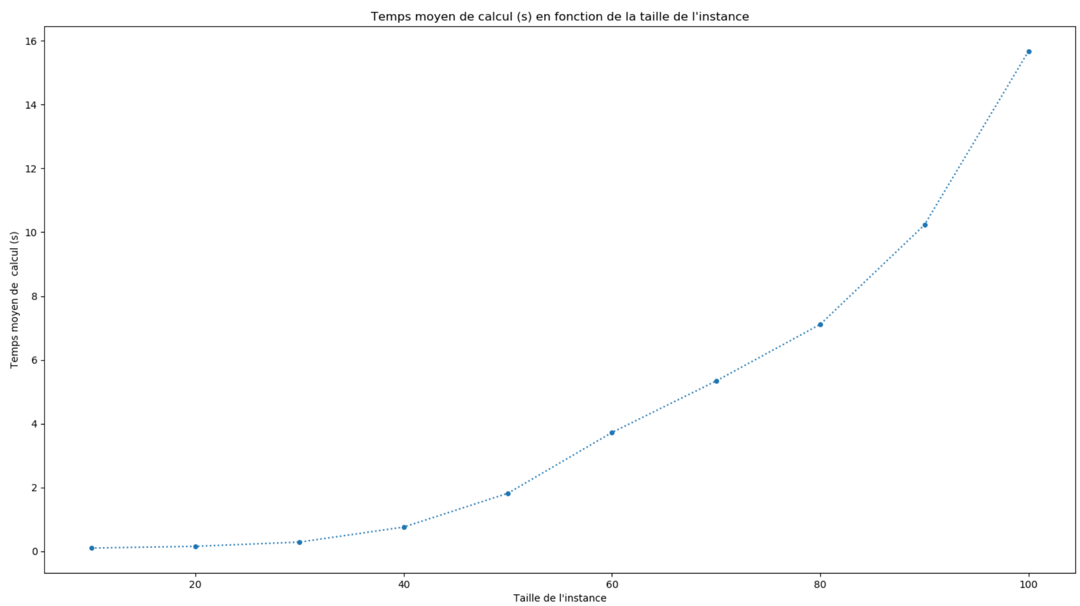

FIGURE 1 – Temps moyen de calcul (s) en fonction de la taille de l'instance

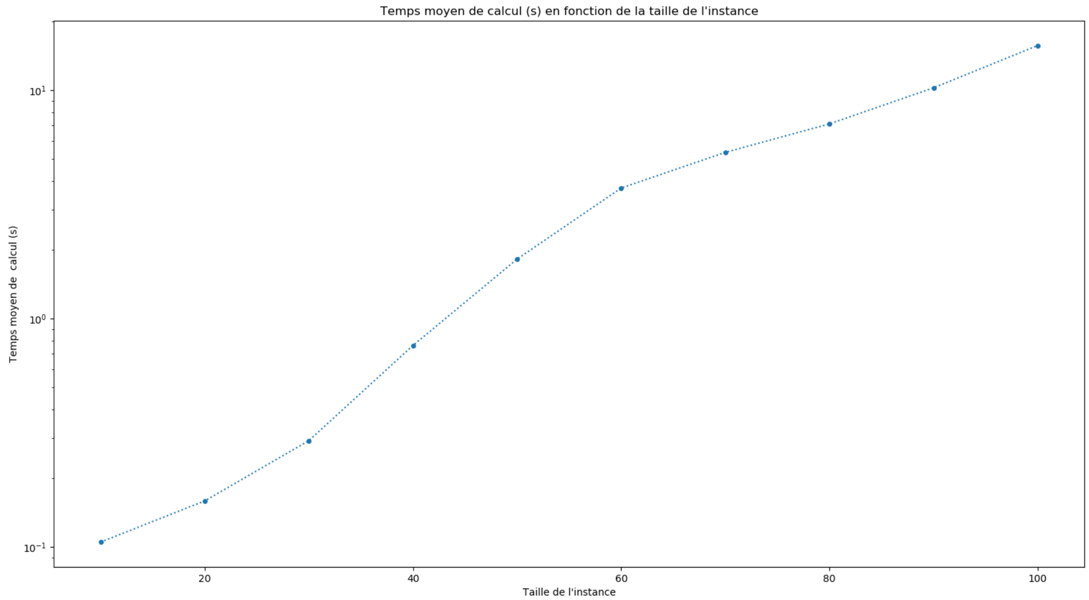

FIGURE 2 – Temps moyen de calcul (s) en fonction de la taille de l'instance

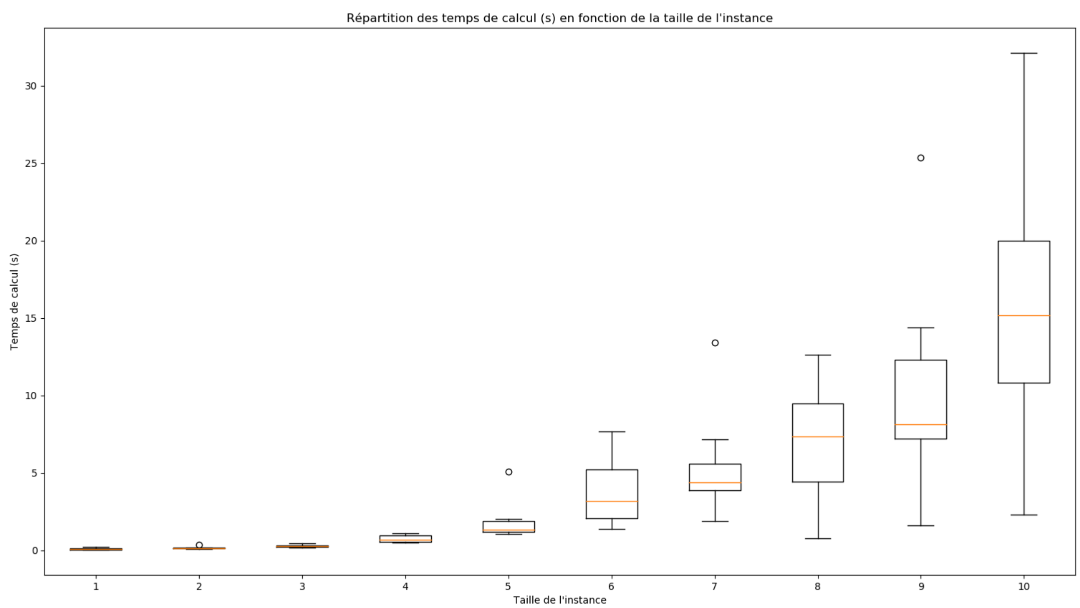

FIGURE 3 – Répartition des temps de calcul (s) en fonction de la taille de l'instance

1.5 Nouveau modèle

2 Projet 2 - Maîtrise des effets néfastes engendrés par la fragmentation su paysage

2.1 Présentation du problème

On souhaite minimiser la fragmentation du paysage, c'est-à-dire l'éclatement de grandes superficies en petites parcelles. Pour cela, on va étudier un indicateur, *la distance moyenne au plus proche voisin (DMPPV)*. On considère un ensemble de parcelles $S = \{s_1, s_2, \dots, s_n\}$ répartis sur un territoire d'aire A.

L'**objectif** est de sélectionner un sous-ensemble de parcelles d'aire totale comprise entre A_{min} et A_{max}, de coût total inférieur ou égal à une certaine valeur B et qui minimise *DMPPV*.

2.2 Modélisation du problème

On modélise le problème avec le programme d'optimisation combinatoire factionnaire suivant :

$$\min \quad z = \frac{\sum_{i_1=1}^{m}\sum_{j_1=1}^{n}\sum_{i_2=1}^{m}\sum_{j_2=1}^{n} d_{i_1 j_1 i_2 j_2} y_{i_1 j_1 i_2 j_2}}{\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}}$$

$$\text{s.c.} \quad A_{min} \leq \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \qquad \textbf{(C1)}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \leq A_{max} \qquad \textbf{(C2)}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} 10 x_{ij} c_{ij} \leq B \qquad \textbf{(C3)}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ijij} = 0 \qquad \textbf{(C4)}$$

$$y_{i_1 j_1 i_2 j_2} \leq x_{i_2 j_2} \qquad \forall i1 \in [\![1;m]\!],\quad \forall j1 \in [\![1;n]\!],\quad \forall i2 \in [\![1;m]\!],\quad \forall j2 \in [\![1;n]\!] \qquad \textbf{(C5)}$$

$$\sum_{i_2=1}^{m}\sum_{j_2=1}^{n} y_{i_1 j_1 i_2 j_2} \leq x_{i_1 j_1} \qquad \forall i1 \in [\![1;m]\!],\quad \forall j1 \in [\![1;n]\!] \qquad \textbf{(C6)}$$

$$x_{ij} \in [\![0;1]\!] \qquad \forall i \in [\![1;m]\!],\quad \forall j \in [\![1;n]\!]$$

$$y_{i_1 j_1 i_2 j_2} \in [\![0;1]\!] \qquad \forall i1 \in [\![1;m]\!],\quad \forall j1 \in [\![1;n]\!],\quad \forall i2 \in [\![1;m]\!],\quad \forall j2 \in [\![1;n]\!]$$

$$A_{min}, A_{max}, B \in N$$

L'étude de ce **programme optimisation** peut se ramener à l'étude du programme linéaire suivant, où l'on pose $f(x,y) = \sum_{i_1=1}^{m}\sum_{j_1=1}^{n}\sum_{i_2=1}^{m}\sum_{j_2=1}^{n} d_{i_1 j_1 i_2 j_2} y_{i_1 j_1 i_2 j_2}$ et $g(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}$:

$$\min \quad z = f(x,y) - \lambda g(x,y)$$

$$
\begin{array}{llr}
\text{s.c.} \quad \mathrm{A}_{min} \leq \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} & & \textbf{(C1)} \\
\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \leq \mathrm{A}_{max} & & \textbf{(C2)} \\
\sum_{i=1}^{m}\sum_{j=1}^{n} 10 x_{ij} c_{ij} \leq \mathrm{B} & & \textbf{(C3)} \\
\sum_{i=1}^{m}\sum_{j=1}^{n} y_{ijij} = 0 & & \textbf{(C4)} \\
y_{i_1 j_1 i_2 j_2} \leq x_{i_2 j_2} & \forall i1 \in [\![1;m]\!], \quad \forall j1 \in [\![1;n]\!], \quad \forall i2 \in [\![1;m]\!], \quad \forall j2 \in [\![1;n]\!] & \textbf{(C5)} \\
\sum_{i_2=1}^{m}\sum_{j_2=1}^{n} y_{i_1 j_1 i_2 j_2} \leq x_{i_1 j_1} & \forall i1 \in [\![1;m]\!], \quad \forall j1 \in [\![1;n]\!] & \textbf{(C6)} \\
x_{ij} \in [\![0;1]\!] & \forall i \in [\![1;m]\!], \quad \forall j \in [\![1;n]\!] & \\
y_{i_1 j_1 i_2 j_2} \in [\![0;1]\!] & \forall i1 \in [\![1;m]\!], \quad \forall j1 \in [\![1;n]\!], \quad \forall i2 \in [\![1;m]\!], \quad \forall j2 \in [\![1;n]\!] & \\
\mathrm{A}_{min}, \mathrm{A}_{max}, \mathrm{B} \in \mathrm{N} & &
\end{array}
$$

On utilise **deux variables de décision** :

— x_{ij} prend 1 si la case de coordonnées (i,j) est sélectionnée, 0 sinon.

— $y_{i_1 j_1 i_2 j_2}$ prend 1 si les cases de coordonnées (i1,j1) et (i2,j2) sont sélectionnée et si (i2,j2) et la case la plus proche de (i1,j1), 0 sinon.

Les **contraintes** ont les significations suivantes :

— Les contraintes **C1** à **C2** formalise le fait que l'**aire totale** des cases sélectionnée doit être comprise entre A_{min} et A_{max}.

— La contrainte **C3** formalise le fait le **coût total des cases sélectionnées** doit être inférieur à un coût B.

— La contrainte **C4** formalise le fait qu'une **case ne peut pas avoir comme case la plus proche elle même**.

— La contrainte **C5** formalise le fait que si une case de coordonnées (i1,j1) est la plus proche d'une autre case (i2,j2), alors (i2,j2) est sélectionnée.

— La contrainte **C6** formalise le fait que si une case (i1,j1) est la plus proche d'une autre case (i2,j2), alors cette case est unique et la case (i1,j1) doit être sélectionnée.

2.3 Résolution du problème

On résout le problème linéaire plus haut en appliquant l'**algorithme de Dinkelbach**. On utilise la matrice des coûts du Tableau 7 et on étudie les 3 cas suivants :

- Cas n°1 : $A_{min} = 30$, $A_{min} = 35$, $B = 920$
- Cas n°2 : $A_{min} = 20$, $A_{min} = 21$, $B = 520$
- Cas n°3 : $A_{min} = 70$, $A_{min} = 75$, $B = 3500$

7	3	10	10	2	8	6	4	5	5
7	7	10	5	2	8	6	3	9	9
7	3	4	6	3	2	4	9	7	8
6	2	7	6	4	7	5	10	7	8
2	4	3	4	9	6	4	9	8	4
7	5	2	9	8	9	5	6	10	10
5	2	3	7	9	9	4	9	6	3
5	2	9	4	2	8	6	9	3	4
9	6	5	4	5	6	8	9	6	6
8	8	7	7	3	5	8	3	9	9

TABLE 7 – Matrice des coûts utilisées

On obtient les résultats visibles sur le Tableau 8, le Tableau 9 et le Tableau 10 qui **correspondent aux résultats attendus**. Un récapitulatif des données voulues est donné dans le Tableau 11.

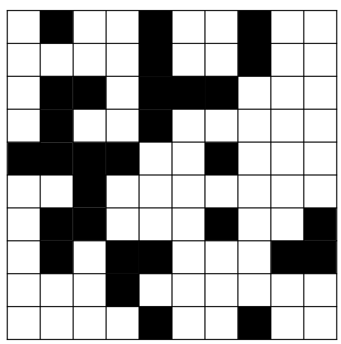

TABLE 8 – Résultats obtenus pour le **Cas n°1** : **nombre d'itérations = 2**, **nombre de noeuds = 0**, **nombre de parcelles sélectionnées = 30** et **DMPPV = 1.155009385**.

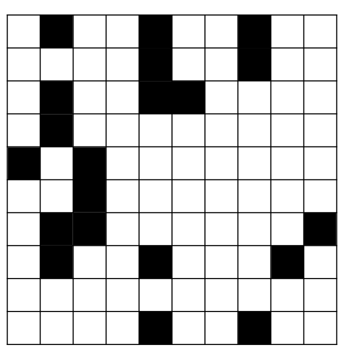

TABLE 9 – Résultats obtenus pour le **Cas n°2** : **nombre d'itérations = 2**, **nombre de noeuds = 0**, **nombre de parcelles sélectionnées = 20** et **DMPPV = 1.2739354335**.

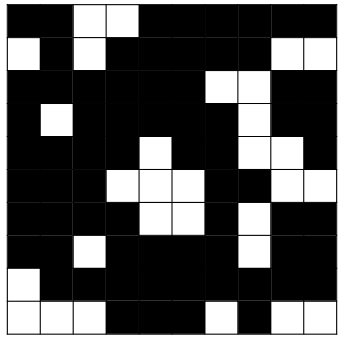

TABLE 10 – Résultats obtenus pour le **Cas n°3** : **nombre d'itérations = 2**, **nombre de noeuds = 0**, **nombre de parcelles sélectionnées = 71** et **DMPPV = 1**.

	temps (s)	nombre de noeuds	nombre d'itérations	DMPPV	nombre de cases sélectionnées
Cas n°1	0,41	0	2	1.155009385	30
Cas n°2	0,35	0	2	1.273935433	20
Cas n°3	0,36	0	2	1	71

TABLE 11 – Résultats obtenus pour les différents cas demandés.

2.4 Étude de l'impact de la taille d'une instance sur le temps moyen de calcul

On étude l'**impact de la taille d'une instance sur le temps moyen de calcul**. Pour cela on crée des instances de manière aléatoire avec le générateur d'instance `generate.py` en suivant les règles suivantes :

— A_{min} est choisi uniformément entre $0.2 * m * n$ et $0.8 * m * n$.

— A_{max} est choisi uniformément entre $A_{min} + 1$ et $A_{max} + 10$.

— B est choisi uniformément entre $10 * m * n$ et $50 * m * n$.

— On fait varier les tailles de 5 en 5 en gardant $m = n$ et en s'arrêtant à 30.

Notre machine de tests dispose de **8 GO de ram** et d'un **processeur i5**. Enfin, pour chaque taille d'instance, on effectue **10 expériences**. On s'intéresse donc au temps **moyen** (Figure 4) ainsi qu'à la répartition des valeurs obtenues (Figure 6) en fonction de la taille de l'instance.

On remarque que **plus la taille des instances est importante, plus le temps de calcul en moyenne est élevé et plus les écarts de temps de calcul sont importants**. L'évolution semble **exponentielle**. En passant au log (Figure 5), on observe que l'évolution est presque linéaire. Il aurait fallut faire plus d'expériences pour avoir un résultat plus précis. On peut tout de même imaginer que le problème est **difficile**.

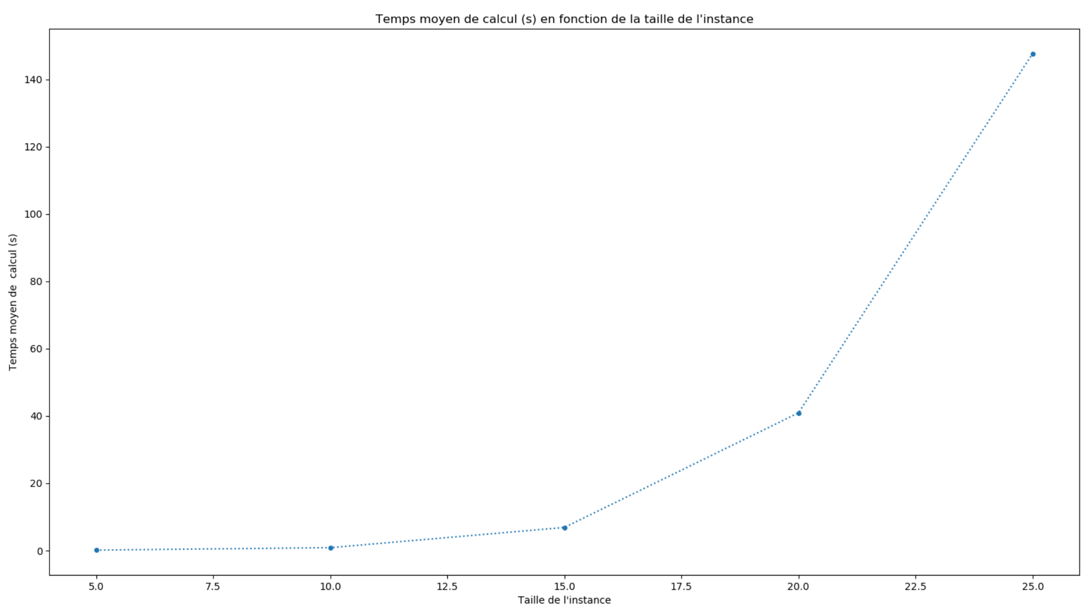

FIGURE 4 – Temps moyen de calcul (s) en fonction de la taille de l'instance

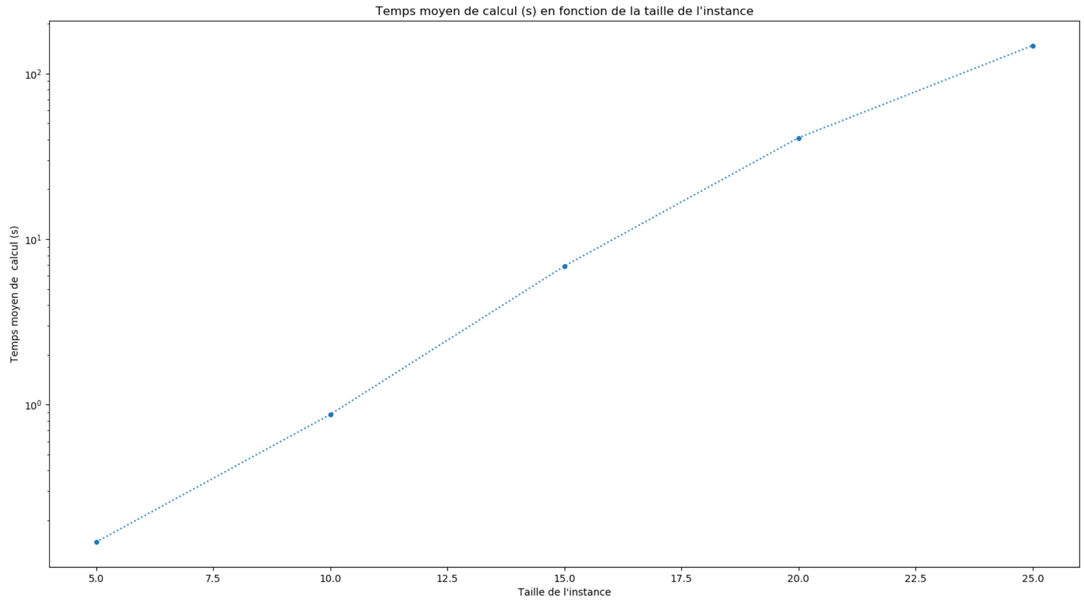

FIGURE 5 – Temps moyen de calcul (s) en fonction de la taille de l'instance

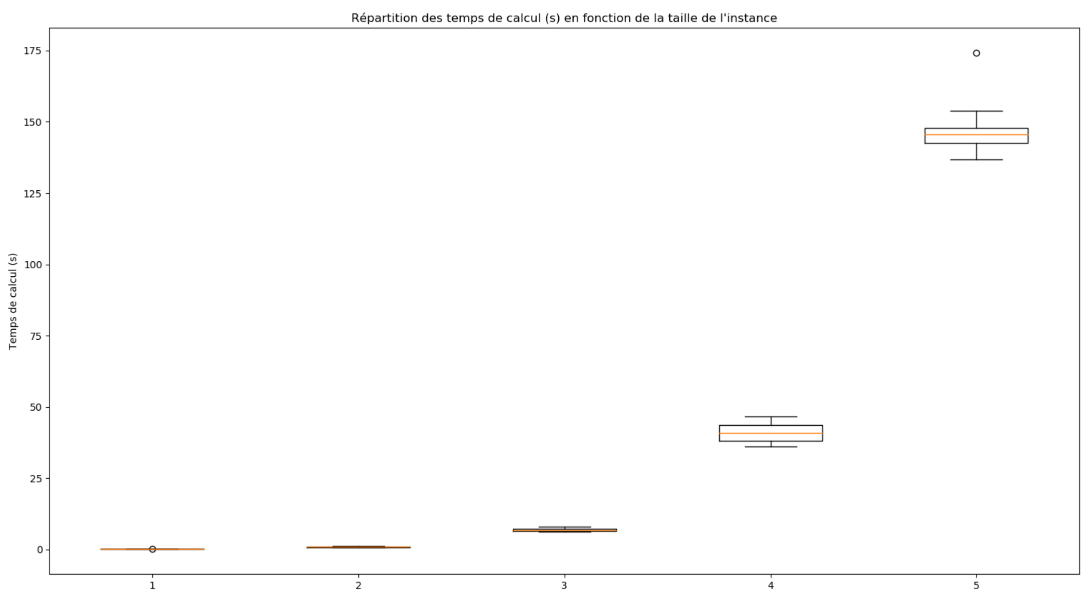

FIGURE 6 – Répartition des temps de calcul (s) en fonction de la taille de l'instance